```
# -*- coding:utf-8 -*-
import tkinter
import tkinter.messagebox

# キャンバスの横方向・縦方向のサイズ（px）
CANVAS_SIZE = 400

# 横方向・縦方向のマスの数
NUM_SQUARE = 8

# 色の設定
BOARD_COLOR = "green" # 盤面の背景色
YOUR_COLOR = "black" # あなたの石の色
COM_COLOR = "white" # 相手の石の色
PLACABLE_COLOR = "yellow" # 次に石を置ける場所を示す色

# プレイヤーを示す値
YOU = 1
COM = 2

class Othello():
    def __init__(self, master):
        '''コンストラクタ'''
        self.master = master # 親ウィジェット
        self.player = YOU # 次に置く石の色
        self.board = None # 盤面上の石を管理する 2 次元リスト
        self.color = { # 石の色を保持する辞書
            YOU : YOUR_COLOR,
            COM : COM_COLOR
        }

        # ウィジェットの作成
        self.createWidgets()

        # イベントの設定
        self.setEvents()

        # オセロゲームの初期化
        self.initOthello()


    def createWidgets(self):
        '''ウィジェットを作成・配置する'''
        # キャンバスの作成
        self.canvas = tkinter.Canvas(
            self.master,
```

```
            bg=BOARD_COLOR,
            width=CANVAS_SIZE+1, # +1は枠線描画のため
            height=CANVAS_SIZE+1, # +1は枠線描画のため
            highlightthickness=0
        )
        self.canvas.pack(padx=10, pady=10)

    def setEvents(self):
        '''イベントを設定する'''
        # キャンバス上のマウスクリックを受け付ける
        self.canvas.bind("<ButtonPress>", self.click)

    def initOthello(self):
        '''ゲームの初期化を行う'''
        # 盤面上の石を管理する２次元リストを作成（最初は全てNone）
        self.board = [[None] * NUM_SQUARE for i in range(NUM_SQUARE)]

        # １マスのサイズ（px）を計算
        self.square_size = CANVAS_SIZE // NUM_SQUARE

        # マスを描画
        for y in range(NUM_SQUARE):
            for x in range(NUM_SQUARE):
                # 長方形の開始・終了座標を計算
                xs = x * self.square_size
                ys = y * self.square_size
                xe = (x + 1) * self.square_size
                ye = (y + 1) * self.square_size

                # 長方形を描画
                tag_name = "square_" + str(x) + "_" + str(y)
                self.canvas.create_rectangle(
                    xs, ys,
                    xe, ye,
                    tag=tag_name
                )

        # あなたの石の描画位置を計算
        your_init_pos_1_x = NUM_SQUARE // 2
        your_init_pos_1_y = NUM_SQUARE // 2
        your_init_pos_2_x = NUM_SQUARE // 2 - 1
        your_init_pos_2_y = NUM_SQUARE // 2 - 1

        your_init_pos = (
            (your_init_pos_1_x, your_init_pos_1_y),
            (your_init_pos_2_x, your_init_pos_2_y)
```

```
        )

        # 計算した描画位置に石（円）を描画
        for x, y in your_init_pos:
            self.drawDisk(x, y, self.color[YOU])

        # 対戦相手の石の描画位置を計算
        com_init_pos_1_x = NUM_SQUARE // 2 - 1
        com_init_pos_1_y = NUM_SQUARE // 2
        com_init_pos_2_x = NUM_SQUARE // 2
        com_init_pos_2_y = NUM_SQUARE // 2 - 1

        com_init_pos = (
            (com_init_pos_1_x, com_init_pos_1_y),
            (com_init_pos_2_x, com_init_pos_2_y)
        )

        # 計算した描画位置に石（円）を描画
        for x, y in com_init_pos:
            self.drawDisk(x, y, self.color[COM])

        # 最初に置くことができる石の位置を取得
        placable = self.getPlacable()

        # その位置を盤面に表示
        self.showPlacable(placable)

    def drawDisk(self, x, y, color):
        '''(x,y)に色がcolorの石を置く（円を描画する）'''
        # (x,y)のマスの中心座標を計算
        center_x = (x + 0.5) * self.square_size
        center_y = (y + 0.5) * self.square_size

        # 中心座標から円の開始座標と終了座標を計算
        xs = center_x - (self.square_size * 0.8) // 2
        ys = center_y - (self.square_size * 0.8) // 2
        xe = center_x + (self.square_size * 0.8) // 2
        ye = center_y + (self.square_size * 0.8) // 2

        # 円を描画する
        tag_name = 'disk_' + str(x) + '_' + str(y)
        self.canvas.create_oval(
            xs, ys,
            xe, ye,
            fill=color,
            tag=tag_name
```

```
139         )
140
141         # 描画した円の色を管理リストに記憶させておく
142         self.board[y][x] = color
143
144     def getPlacable(self):
145         '''次に置くことができる石の位置を取得'''
146         placable = []
147
148         for y in range(NUM_SQUARE):
149             for x in range(NUM_SQUARE):
150                 # (x,y) の位置のマスに石が置けるかどうかをチェック
151                 if self.checkPlacable(x, y):
152                     # 置けるならその座標をリストに追加
153                     placable.append((x, y))
154
155         return placable
156
157     def checkPlacable(self, x, y):
158         '''(x,y)に石が置けるかどうかをチェック'''
159         # その場所に石が置かれていれば置けない
160         if self.board[y][x] != None:
161             return False
162
163         if self.player == YOU:
164             other = COM
165         else:
166             other = YOU
167
168         # (x,y)座標から縦横斜め全方向に対して相手の意思が裏返せるかどうかを確認
169         for j in range(-1, 2):
170             for i in range(-1, 2):
171
172                 # 次の方向の確認に移る
173                 if i == 0 and j == 0:
174                     continue
175
176                 # その方向が盤面外になる場合も次の方向の確認に移る
177                 if x + i < 0 or x + i >= NUM_SQUARE or y + j < 0 or y + j >= NUM_SQUARE:
178                     continue
179
180                 # 隣が相手の色でなければその方向に石を置いても裏返せない
181                 if self.board[y + j][x + i] != self.color[other]:
182                     continue
183
```

```
184             # 置こうとしているマスから遠い方向へ１マスずつ確認
185             for s in range(2, NUM_SQUARE):
186                 # 盤面外のマスはチェックしない
187                 if x + i * s >= 0 and x + i * s < NUM_SQUARE and y + j * s >= 0 and y + j * s < NUM_SQUARE:
188 
189                     if self.board[y + j * s][x + i * s] == None:
190                         # 自分の石が見つかる前に空きがある場合
191                         # この方向の石は裏返せないので次の方向をチェック
192                         break
193 
194                     # その方向に自分の色の石があれば石が裏返せる
195                     if self.board[y + j * s][x + i * s] == self.color[self.player]:
196                         return True
197 
198     # 裏返せる石がなかったので(x,y)に石は置けない
199     return False
200 
201 def showPlacable(self, placable):
202     '''placableに格納された次に石が置けるマスの色を変更する'''
203     for y in range(NUM_SQUARE):
204         for x in range(NUM_SQUARE):
205 
206             # fillを変更して石が置けるマスの色を変更
207             tag_name = "square_" + str(x) + "_" + str(y)
208             if (x, y) in placable:
209                 self.canvas.itemconfig(
210                     tag_name,
211                     fill=PLACABLE_COLOR
212                 )
213 
214             else:
215                 self.canvas.itemconfig(
216                     tag_name,
217                     fill=BOARD_COLOR
218                 )
219 
220 def click(self, event):
221     '''盤面がクリックされた時の処理'''
222     if self.player != YOU:
223         # COMが石を置くターンの時は何もしない
224         return
225 
226     # クリックされた位置がどのマスであるかを計算
227     x = event.x // self.square_size
228     y = event.y // self.square_size
229 
```

```
230         if self.checkPlacable(x, y):
231             # 次に石を置けるマスであれば石を置く
232             self.place(x, y, self.color[self.player])
233 
234     def place(self, x, y, color):
235         '''(x,y)に色がcolorの石を置く'''
236         # (x,y)に石が置かれた時に裏返る石を裏返す
237         self.reverse(x, y)
238 
239         # (x,y)に石を置く（円を描画する）
240         self.drawDisk(x, y, color)
241 
242         # 次に石を置くプレイヤーを決める
243         before_player = self.player
244         self.nextPlayer()
245 
246         if before_player == self.player:
247             # 前と同じプレイヤーであればスキップされたことになるのでそれを表示
248             if self.player != YOU:
249                 tkinter.messagebox.showinfo("結果", "あなたのターンをスキップしました")
250             else:
251                 tkinter.messagebox.showinfo("結果", "COMのターンをスキップしました")
252 
253         elif not self.player:
254             # 次に石が置けるプレイヤーがいない場合はゲーム終了
255             self.showResult()
256             return
257 
258         # 次に石がおける位置を取得して表示
259         placable = self.getPlacable()
260         self.showPlacable(placable)
261 
262         if self.player == COM:
263             # 次のプレイヤーがCOMの場合は1秒後にCOMに石を置く場所を決めさせる
264             self.master.after(1000, self.com)
265 
266     def reverse(self, x, y):
267         '''(x,y)に石が置かれた時に裏返す必要のある石を裏返す'''
268         if self.board[y][x] != None:
269             # (x,y)にすでに石が置かれている場合は何もしない
270             return
271 
272         if self.player == COM:
273             other = YOU
274         else:
275             other = COM
```

```
276
277         for j in range(-1, 2):
278             for i in range(-1, 2):
279                 # 真ん中方向はチェックしてもしょうがないので次の方向の確認に移る
280                 if i == 0 and j == 0:
281                     continue
282
283                 if x + i < 0 or x + i >= NUM_SQUARE or y + j < 0 or y + j >= NUM_SQUARE:
284                     continue
285
286                 # 隣が相手の色でなければその方向で裏返せる石はない
287                 if self.board[y + j][x + i] != self.color[other]:
288                     continue
289
290                 # 置こうとしているマスから遠い方向へ１マスずつ確認
291                 for s in range(2, NUM_SQUARE):
292                     # 盤面外のマスはチェックしない
293                     if x + i * s >= 0 and x + i * s < NUM_SQUARE and y + j * s >= 0 and y + j * s < NUM_SQUARE:
294
295                         if self.board[y + j * s][x + i * s] == None:
296                             # 自分の石が見つかる前に空きがある場合
297                             # この方向の石は裏返せないので次の方向をチェック
298                             break
299
300                         # その方向に自分の色の石があれば石が裏返せる
301                         if self.board[y + j * s][x + i * s] == self.color[self.player]:
302                             for n in range(1, s):
303
304                                 # 盤面の石の管理リストを石を裏返した状態に更新
305                                 self.board[y + j * n][x + i * n] = self.color[self.player]
306
307                                 # 石の色を変更することで石の裏返しを実現
308                                 tag_name = "disk_" + str(x + i * n) + '_' + str(y + j * n)
309                                 self.canvas.itemconfig(
310                                     tag_name,
311                                     fill=self.color[self.player]
312                                 )
313
314                             break
315
316     def nextPlayer(self):
317         '''次に石を置くプレイヤーを決める'''
318         before_player = self.player
319
```

```
320         # 石を置くプレイヤーを切り替える
321         if self.player == YOU:
322             self.player = COM
323         else:
324             self.player = YOU
325 
326         # 切り替え後のプレイヤーが石を置けるかどうかを確認
327         placable = self.getPlacable()
328 
329         if len(placable) == 0:
330             # 石が置けないのであればスキップ
331             self.player = before_player
332 
333             # スキップ後のプレイヤーが石を置けるかどうかを確認
334             placable = self.getPlacable()
335 
336             if len(placable) == 0:
337                 # それでも置けないのであれば両者とも石を置けないということ
338                 self.player = None
339 
340     def showResult(self):
341         '''ゲーム終了時の結果を表示する'''
342         # それぞれの色の石の数を数える
343         num_your = 0
344         num_com = 0
345 
346         for y in range(NUM_SQUARE):
347             for x in range(NUM_SQUARE):
348                 if self.board[y][x] == YOUR_COLOR:
349                     num_your += 1
350                 elif self.board[y][x] == COM_COLOR:
351                     num_com += 1
352 
353         # 結果をメッセージボックスで表示する
354         tkinter.messagebox.showinfo("結果", "あなた" + str(num_your) + "：COM" + str(num_com))
355 
356     def com(self):
357         '''COMに石を置かせる'''
358         # 石が置けるマスを取得
359         placable = self.getPlacable()
360 
361         # 最初のマスを次に石を置くマスとする
362         x, y = placable[0]
363 
```

```
364         # 石を置く
365         self.place(x, y, COM_COLOR)
366 
367 # スクリプト処理ここから
368 app = tkinter.Tk()
369 app.title('othello')
370 othello = Othello(app)
371 app.mainloop()
372 
```